ASSESSMENT 2:

SENTIMENT ANALYSIS & NAMED ENTITY RECOGNITION IMPLEMENTATION ON ADVERSE DRUG REACTION CONTEXT

TEAM: Nerveless | Date 24/05/2021

[COSC2753] MACHINE LEARNING

Lecturer: Dr. Dang Pham Thien Duy

Members:

1. Le Nguyen Minh Huy - s3777280
2. Nguyen Manh Triet – s3678932

Table of Contents

Tables of Figures

Tables of Tables

1. Introduction

The report aims to demonstrate several specialized techniques implemented in the NLP (Natural Language Processing) task, particularly text recognition and named entity recognition related to Adverse Drug Reaction context, and their results conducted within the project. A detailed evaluation and proofs are provided to clarify the outcomes. Moreover, the report has analysed the results of the final model in the ultimate judgment part.

Dataset

The datasets implemented in both models are extracted from the Dataset Card for Adverse Drug Reaction Data v2 via Hugging Face's dataset API. The classification model utilizes the *Ade_corpus_v2_classification* instance, which is used to classify whether a sentence is ADE-related (True) or not (False), containing 33709 rows and 2 attributes. The NER model is trained on the *Ade_corpus_v2_drug_dosage_relation* instance, which is used to provide the index position of drug and dosage word within a sentence, containing 279 rows and 4 attributes.

Regarding **the text classification dataset**, after conducting the Exploratory Dataset Analysis, it is clear that there is an imbalance between two classes of the target column label in *the Ade_corpus_v2_classification* dataset. The graph below illustrates about 71% of the medical context is classified as 0 (not related), and only the remaining 29% are labelled 1 (related).

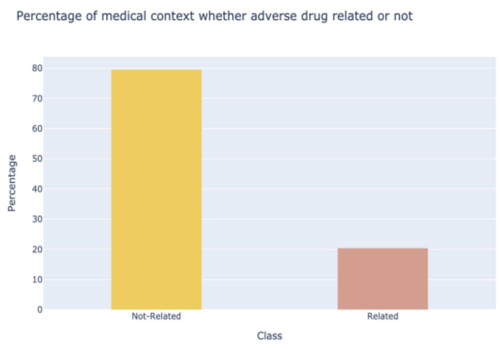

Figure 1: Imbalanced class in Ade_corpus_v2_classificaiton dataset.

According to Yogesh, training a model in an imbalanced dataset leads to a significant bias towards larger classes. Moreover, the class with fewer data points will be treated as noise and ignored during the training process [1]. There are two main ways to balancing the datasets: oversample and undersampling. As we lose much information using the undersampling method, the team implemented oversampling technique which helps them to have more data of the minority class. In NLP, the back-translation technique allows us to paraphrase each instance in the dataset. Specifically, we did translate the original corpus from English to three common languages that are French, Japanese, and Spanish, and then back translating them to English. To implement the technique, we have used the **BackTranslation** python package which utilizes the googletrans library and Baidu Translation API. The detailed process can be viewed within the Notebook.

The *Ade_corpus_v2_drug_dosage_relation* dataset was modified to a suitable data format, from converting to spaCy's BILOU data scheme to IOB tagging format. The detailed information can be viewed within the Notebook. Concerning the dataset after pre-processing, there is an overwhelming imbalance between each tag category. Based on the plotted diagram below, it is clear that the dataset is heavily biased towards the O-tag, which partakes over 80% of the total tags, while the others only contribute a small proportion.

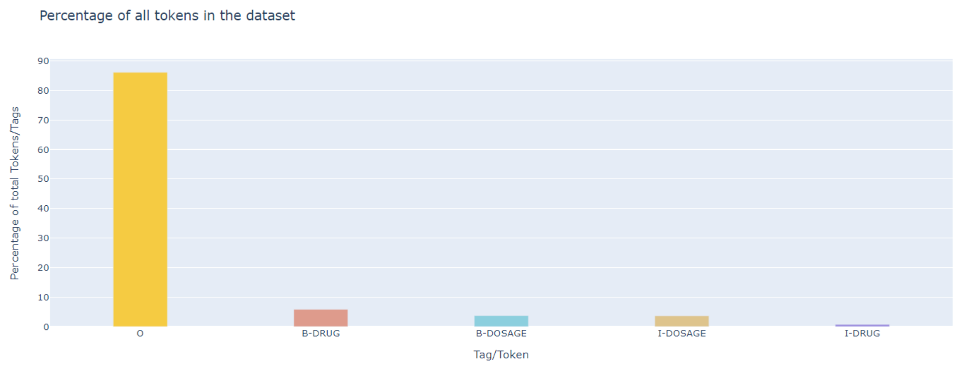

Figure 2: Percentage of all tokens.

To address this inadequate distribution of tags, we decide to evaluate our model using **the Macro-F1** score, which is performed by first computing the F1 score per class/label and then averaging them. One of the research papers about NER, proposed by Tomanek, has mentioned that the NER model was leveraging with macro-f1 score due to the imbalance issue [2]. While the F1 score is a good evaluation metric for the binary classification problem, Macro Average F1 is not only efficient for multi-class classification but also preferable if there is an imbalanced class. When macro-averaging, all attributes contribute equally regardless of how often they appear in the dataset.

2. Approach

2.1 Machine Learning Approach

2.1.1 Text Classification task

The team decided to use **TFIDFVectorizer** which enables us to emphasize the importance of words in specific documents and their relevance in other documents as well.

b. Feature extraction

TFIDF stands for Term Frequency - Inverse Document Frequencies, which not only aims to indicate the importance of the words in the given context but also takes into account the relation to other documents from the same corpus. The algorithm is focusing on the number of times that word appears in a document and other documents in the corpus as well. To be more specific, a word that often appears in a document has more relevance for that document, which interprets there is a higher probability that the document is about that specific word. Moreover, when a word frequently appears in lots of different documents, it would be considered as a less-important word since it is relevant for many documents from the whole set.

c. Modelling

By applying TFIDFVectorizer, each document in the given corpus will be represented as a vector containing the weight of words. Then, the vectors will be fed into three ML models for comparison based on F1 score.

- MultinomialNB: MultinomialNB stands for Multinomial Naive Bayes algorithm which is a probabilistic learning method often applied in Natural Language Processing (NLP). Based on the Bayes theorem, the algorithm returns the label of a given text. Specifically, it computes the probability of each label for a particular document and then chooses the result of the highest probability as output.
- LightGBM: LightGBM is a gradient boosting which is currently mostly used in classification tasks. Regarding the tree-based learning algorithm, the LightGBM model grows trees vertically instead of horizontally like others, which is called tree leaf-wise. By using this algorithm, we did reduce significant loss than other level-wise algorithms as well as choose a max-delta-loss leaf to grow. Moreover, the LightGBM algorithm can work fast with a large amount of data and takes lower memory to execute.
- Logistic Regression: Logistic regression uses the logistic function known as the sigmoid function. Logistic regression can not only predict the discrete label but also predict the probabilities for each class based on given attributes. The sigmoid function is an S-shaped curve that can take any real number and map it into a value between 0 and 1.

2.1.2 Name Entity Recognition task

We implemented multiple strategies in this phase, each was an improved version of the previous one, in order to acquire the best Machine Learning model.

- Our baseline approach was to construct a simple classifier by inheriting scikit-learn base classes **BaseEstimator** and **TransformerMixin**, to implement such functions like **get_params, get_params** and **fit_transform.** The result was overall acceptable, but this is just a
- To address the issue, the team decided to convert words to simple feature vector then apply RandomForestClassifier model to recognize entities of the words and make prediction accordingly. We hence applied a more sophisticated feature extraction function based upon the,
- Further model performance improvement was accomplished by employing **Conditional Random Field (CRF)** model via **sklearn-crfsuite** and **ELI5.** The idea was to implement sequence labelling technique to predict the sequences that use the contextual information to add information which will be used by the model to make a correct prediction.

2.2 Deep Learning Approach

Recurrent Neural Network (RNN) as described is a powerful and robust type of neural network in Natural Language Processing (NLP). The model itself is recurrent in nature as it performs the same function for every input of data while the output of the current input relies on the previous one computation. Right after being produced, the output is then copied and returned to the recurrent network. Together with the current input, the model's decision-making capability relies upon the output that has been learned from the preceding input [3].

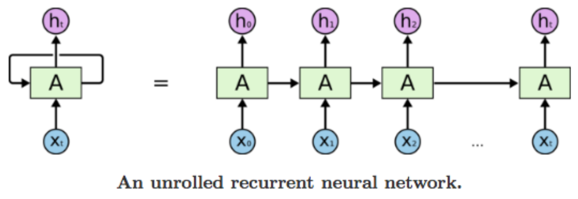

Figure 3: How recurrent neural network works [3].

However, due to the gradient vanishing phenomenon, RNN fails to process long sentences, which is a critical issue in our project [4]. Therefore, to tackle this problem, the team decided to use the Long Short-Term Memory (LSTM) model - a modified version of RNN instead **for both tasks.** The LSTM model can control the flow of information through a memory cell c in the current network with three gates: input gate i, forgotten gate f, and output gate o. More concretely, the input of the cells will be processed by the activation function of the input gate, the output is multiplied by the output gate and the previous cell values will be multiplied by the forget gate.

a. Tokenization and Encoding

Firstly, all sentences from both datasets are converted into vectors of words. After that, the number of words was applied to the Tokenizer to build a vocabulary and fit the training set. Then, the arrays containing words generated from both training and validation sets were encoded and turned into the integer sequence.
On the other hand, to ensure that all sequences in the corpus have the same length, which enables the model to be fitted with input data, the team used the ***padding sequence*** function. The key idea of padding sequence is padding zero (0) to the beginning or end of each document until each document has the same length as the longest one [5]. In this situation, the team has configured the hyperparameter *padding* of the *pad_sequences* function to "post", which means all zero will be added to the ending of the integer sequences shorter than the value of max_len parameter, otherwise, sequences longer than that value was truncated. However, as truncating the sequences would lose lots of valuable information, the team decided to define the max_len value equals to the maximum length of a sentence among the corpora. When accomplishing the encode process and padding, the dimensionality of the training set, as well as validation, would be 2D integer tensors.

d. Word Embedding

Before adapting the training set to the neural network, it is essential to have an embedding layer which enables us to convert each word into a fixed length vector. The embedding layers are defined as a lookup table in which words are the keys and the dense word vectors are the values [6]. To be more specific, a 12-dimensional embedding for all unique words in the training set was added to the network, which is indicated by the value of output_dim parameter. As a result, the 2D integer vectors were generated to 3D tensors of shape (None, maxlen, 12), which are both able to be processed by CNN and RNN layer.

e. Modelling

Long Short-Term Memory (LSTM)

The team has done several experiments with LSTM models for **text classification** task. First of all, we implement the RNN model with a single LSTM layer without adding a dropout layer. Then, we started adding additional LSTM layers and applied the dropout technique in an attempt to increase the model's performance based on evaluation metrics, F1 score. During training, at every iteration, the dropout layer randomly selects several nodes along with their incoming and outgoing connections from the neural network. This leads to a prevention from adapting too much. [7] According to Shubham Jain, the technique significantly reduces overfitting and achieves substantial improvements over other regularization methods.[8]

- One-layer LSTM: An RNN model with only one LSTM layer and two activation functions: *ReLU* and *sigmoid*.
- One-layer LSTM with dropout: For addressing overfitting issue, a regularization technique - dropout was applied to the one-LSTM-layer model. The team has chosen the dropout rate of 0.5.
- Two-layer LSTM with dropout: The related work has proved that RNN model performance can improve thanks to the number of LSTM layers, in other words the depth [9]. Hence, the team decided to experiment with two LSTM layers with the same dropout rate which is 0.5.
- Three-layer LSTM with dropout: Inspired by the high performance of Google Translate tool using seven layers of LSTM [10], the team stacked their model with another LSTM layer as well as the dropout layer.
- Bidirectional LSTM with dropout: The team replaced one LSTM layer with one Bidirectional LSTM (BLSTM) layer. Although BLSTM has double memory cells compared to the LSTM layer, the text data is processed in both directions, which helps the model capture the patterns that may be fail recognized in a unidirectional LSTM.

For **Named Entity Recognition** task, plethora methods were documented by using Bi-LSTM as a backbone. Primarily, we experimented with a baseline Bi-LSTM model with embedded dropout_rate = 0.5. After that, we proceeded to tweak more parameters in Bi-LSTM layers such as recurrent_dropout, embedding_size and apply stacked LSTM layers to increase the model learning capability.

- Bi-LSTM baseline with dropout: by using Bi-LSTM instead of the traditional one, the model is able to preserve context information in both past and future, comparing to LSTM which only capable to intake past information. We also added one dropout layer having rate of 0.1. In the Bi-LSTM layer, dropout layer is a regularization technique to avoid overfitting.
- Bi-LSTM + one more LSTM layer with dropout: to help the model capturing more information, we added up 1 more LSTM layer.

Convolutional Neural Network (CNN)

CNNs have been applied to several problems in NLP and gotten remarkable results recently. Specifically, such task like **sentiment analysis** was enhanced in terms of the accuracy and processing time by using CNN model [11]. As a result, we adopt CNN as one of solutions for the depicted task. Various hyperparameters testings were performed to conclude the state-of-the-art structure for text classification task:

1. CNN with dropout: The team has implemented the model with one convolutional layer followed by a max-pooling layer and a dropout layer having rate of 0.5 to address the overfitting problem.
2. Deep CNN: This model contains two sets of convolutional layers followed by a max pooling layer. Moreover, after the second max pooling layer, the team has included a dropout function of 0.5 rate to prevent overfitting problem. Then, a Dense layer as known as fully connected layer was appended with the dropout function. The final layer is a single Dense layer with a SoftMax activation function.

3. Ultimate judgement

Experiment result of text classification model

Table 1: Experiment results of medical text classification task.

Approach	Type of Network	Model	F1 score
Machine Learning		MultinomialNB	0.857

		LightGBM	0.828
		Logistic Regression	0.867
Deep learning	RNN	One LSTM layer (baseline)	0.771
		One LSTM layer with dropout	0.862
		Two LSTM layers with dropout	0.866
		Three LSTM layers with dropout	0.854
		Bidirectional LSTM with dropout	0.864
	CNN	**CNN with dropout**	**0.870**
		Deep CNN	0.840

The table above has summarized the main results of all approaches implementing for **medical text classification** problem. To evaluate the model on the validation set, the team chose the metrics of F1 score, a harmonic means of precision and recall. Since F1 score conveys the balance between precision and recall, it is preferable to use in some classification problem. Regarding the first approach, Logistic Regression gets the highest score among three Machine Learning model, 0.867 compared to 0.857 and 0.828. This model achieves a remarkable result when compared to five experiments with LSTM Deep Learning model. On the other hand, although CNN is not widely used in NLP tasks, our **CNN with dropout** model has been considered as the best one with F1 score of 0.87. The result suggests a great promise of adapting a **CNN model** in NLP field as well as balancing dataset using **back-translation**.

Experiment result of named entity recognition model

Table 2: Experiment results of Named Entity Recognition tasks.

Approach	Type of network	Model	Macro F1 score
Machine Learning		BaseEstimator + TransformerMixin classes	0.59
		Random Forest	0.25
		CRF	0.664
		CRF_tuned	**0.677**
Deep Learning	RNN	Bi-LSTM with dropout	0.30
		Bi-LSTM + LSTM with dropout	0.31

The table has encapsulated all methods applying for **named entity recognition** task. For this task, as addressed earlier, Macro-F1 score will be our evaluation metric as it will give the same importance to each label. In terms of Machine Learning approach, the fine-tuned CRF model achieve the highest result among other counterparts. This model surpassed our expectation as it foreshadowed Bi-LSTM model, which is well-known for its performance in the NER task. Despite its mediocre performance, Bi-LSTM is still one of the most sufficient techniques of the NER task if we can acquire bigger dataset or properly handle imbalance issue with suitable techniques.

4. References

[1] Y. Kothiya., "How I handled imbalanced text data." towardsdatascience. https://towardsdatascience.com/how-i-handled-imbalanced-text-data-ba9b757ab1d8 (accessed May 18, 2021).

[2] K. Tomanek., and H. Udo. “Reducing Class Imbalance during Active Learning for Named Entity Annotation,” in *Proc. 2009 Int. Conf. Knowledge capture*, pp. 105-112. doi:10.1145/1597735.1597754.

[3] A. Mittal., “Understanding RNN and LSTM.” Medium. https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e (accessed May 18, 2021).

[4] N. Donges., “A Guide To RNN: Understanding Recurrent Neural Networks And LSTM.” builtin. https://builtin.com/data-science/recurrent-neural-networks-and-lstm (accessed May 18, 2021).

[5] “tf.keras.preprocessing.sequence.pad_sequences.” TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences (accessed May 18, 2021).

[6] S. Saxena., “Understanding Embedding Layer in Keras.” medium. https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce#:~:text=Embedding%20layer%20enables%20us%20to,way%20along%20with%20reduced%20dimensions. (accessed May 17, 2021).

[7] S. Nitish., H. Geoffrey., K. Alex., S. Ilya., and S. Ruslan. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” Journal of Machine Learning Research, Jan 2014.

[8] J. Shubham., “An Overview of Regularization Techniques in Deep Learning (with Python code).” https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/ (accessed May 19, 2021).

[9] C. Alexis., S. Holger., B. Loıc., and L. Yann. “Very deep convolutional networks for natural language processing”. arXiv preprint, 2016.

[10] F. Chollet. Deep learning with python. Manning Publications Co., 2017.

[11] M. Ari Nasichuddin, T. Bharata Adji, and W. Widyawan. “Performance Improvement Using CNN for Sentiment Analysis,” International Journal of Information Technology and Electrical Engineering, vol. 2, no. 1, doi:10.22146/ijitee.36642,